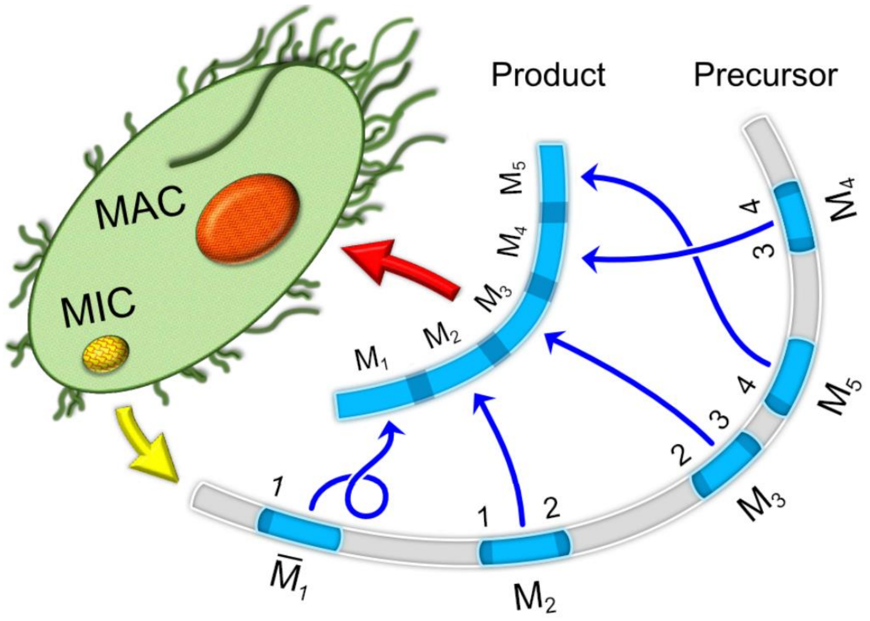
Applications of Persistent Homology

MUSTAFA HAJIJ

Part I
Clustering

Cilliate

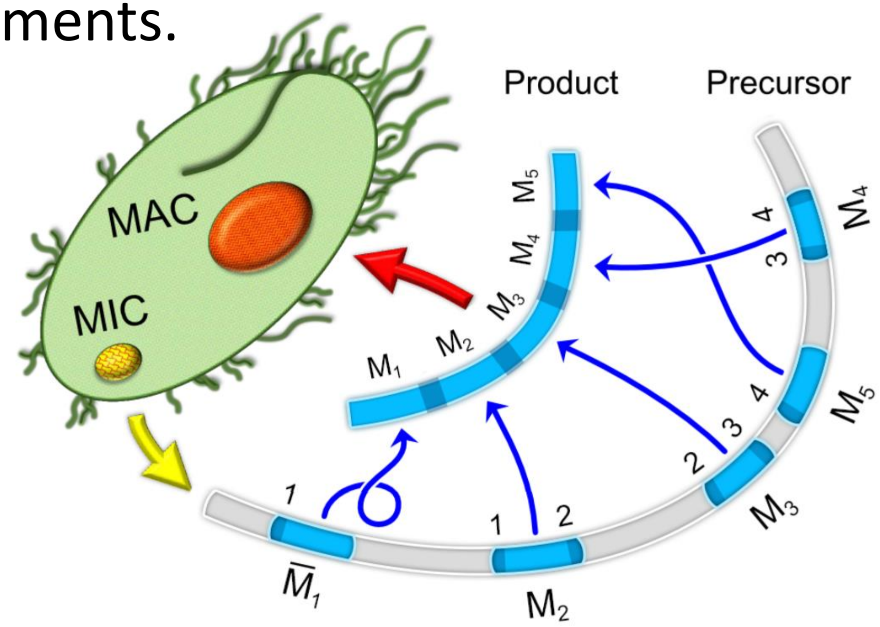
Oxytricha trifallax, a species of ciliate, undergoes massive genome rearrangements during the development of a macronucleus (MAC) from a micronucleus (MIC).



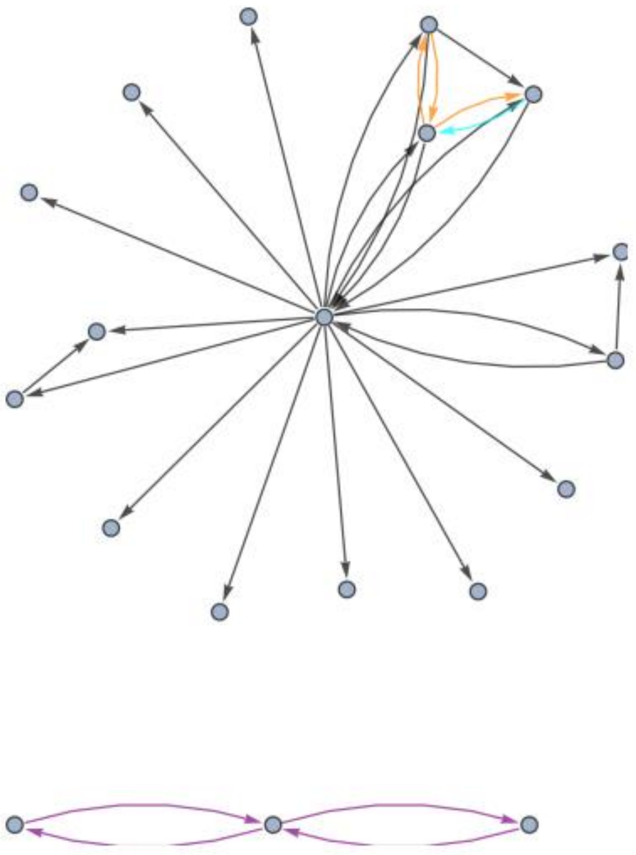
Cilliate

Oxytricha trifallax, a species of ciliate, undergoes massive genome rearrangements during the development of a macronucleus (MAC) from a micronucleus (MIC).

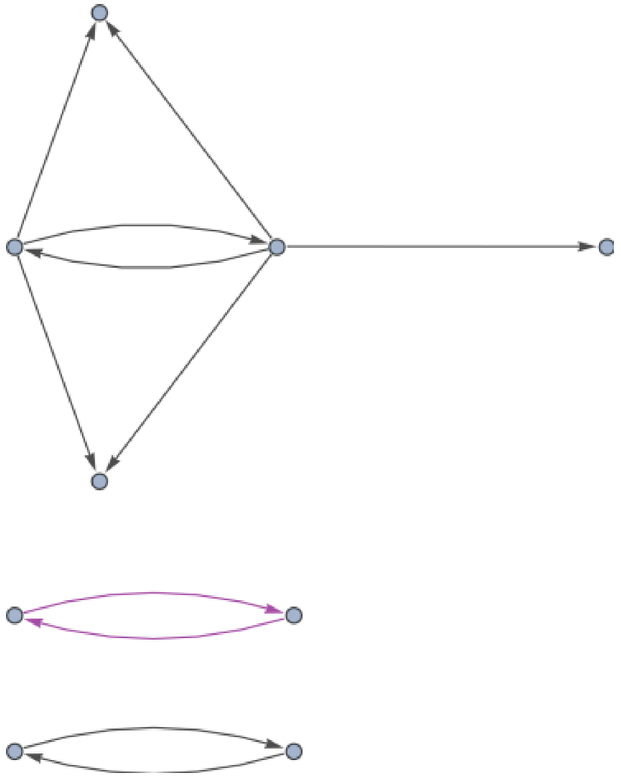
It is used as a model organism to study DNA rearrangements.



Examples

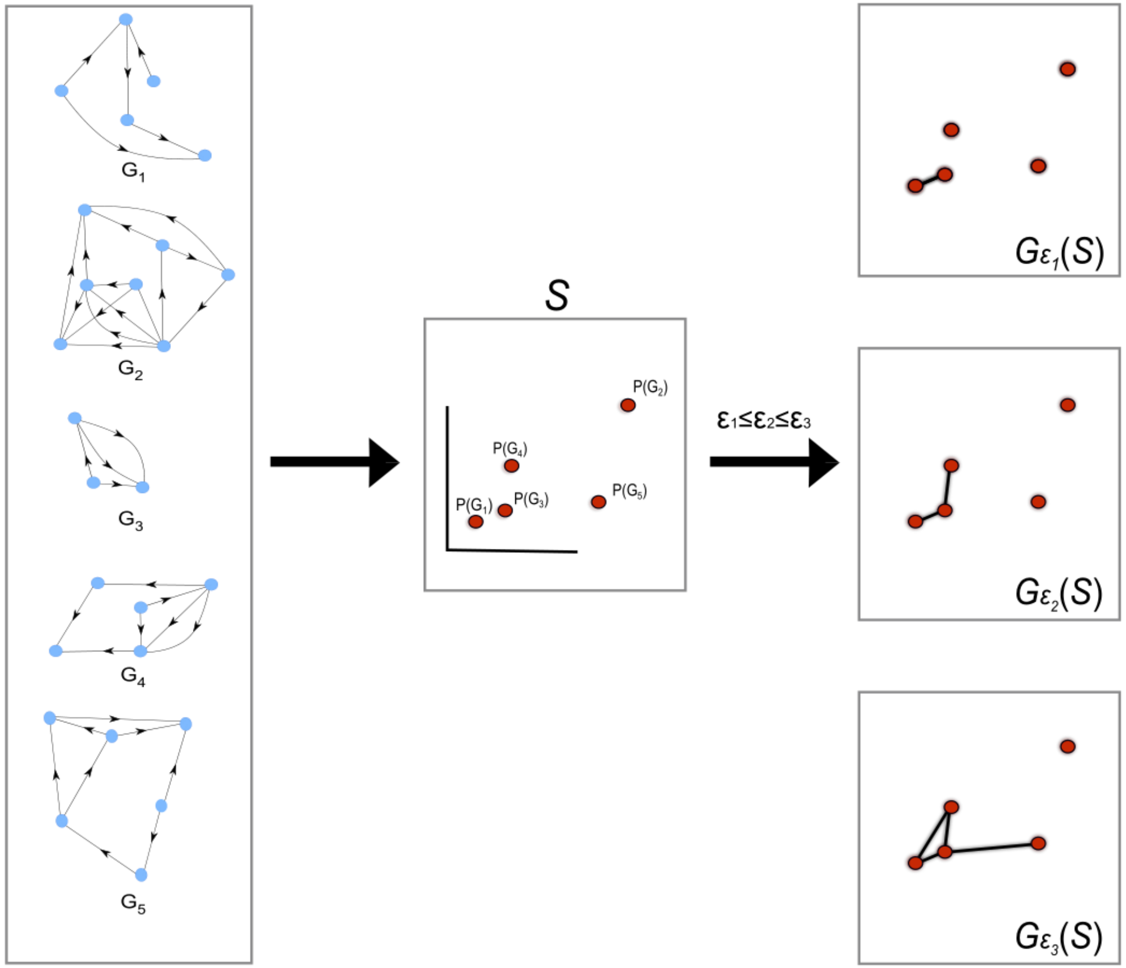


ctg7180000088096



ctg7180000069209

From Graphs to a Point Cloud To Filtration

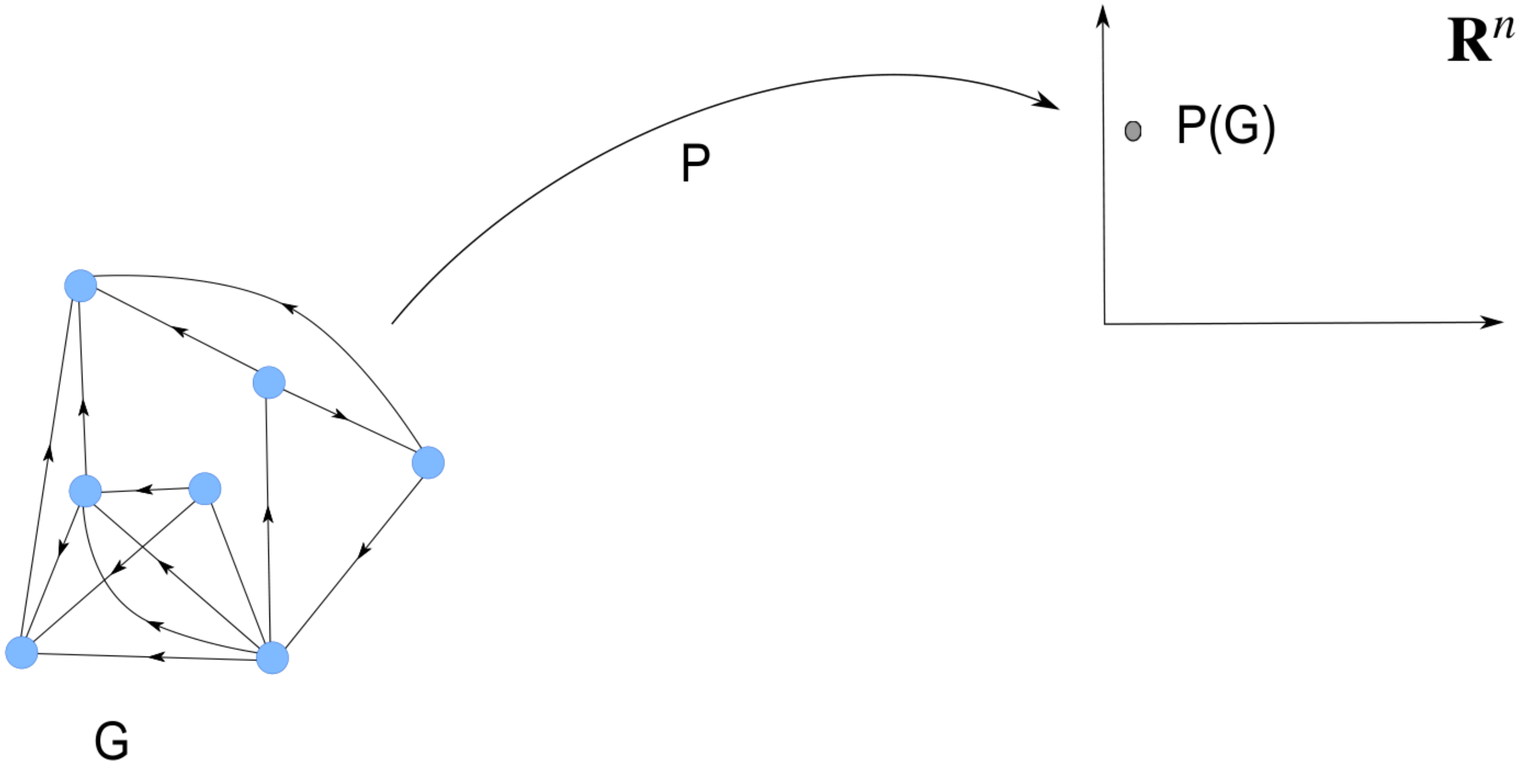


the set of graphs that represents the contigs.

Represent graphs as points in a Euclidean space.

Filtration

From Graphs to a Point Cloud



Associate to every graph G a feature vector P(G), a point in a Euclidean space, that represents the graph G.

From Graphs to a Point Cloud

The vector P(G) is defined as follows.

Global Features Vector: Pg (G) =< V(G), E(G), CN(G) >
V(G) : # of vertices,
E(G) # of edges in Pg (G)
CN(G) : the size of the largest clique in G.

Valence Features Vector: Pv (G) : the valency of the vertices ordered decreasingly.

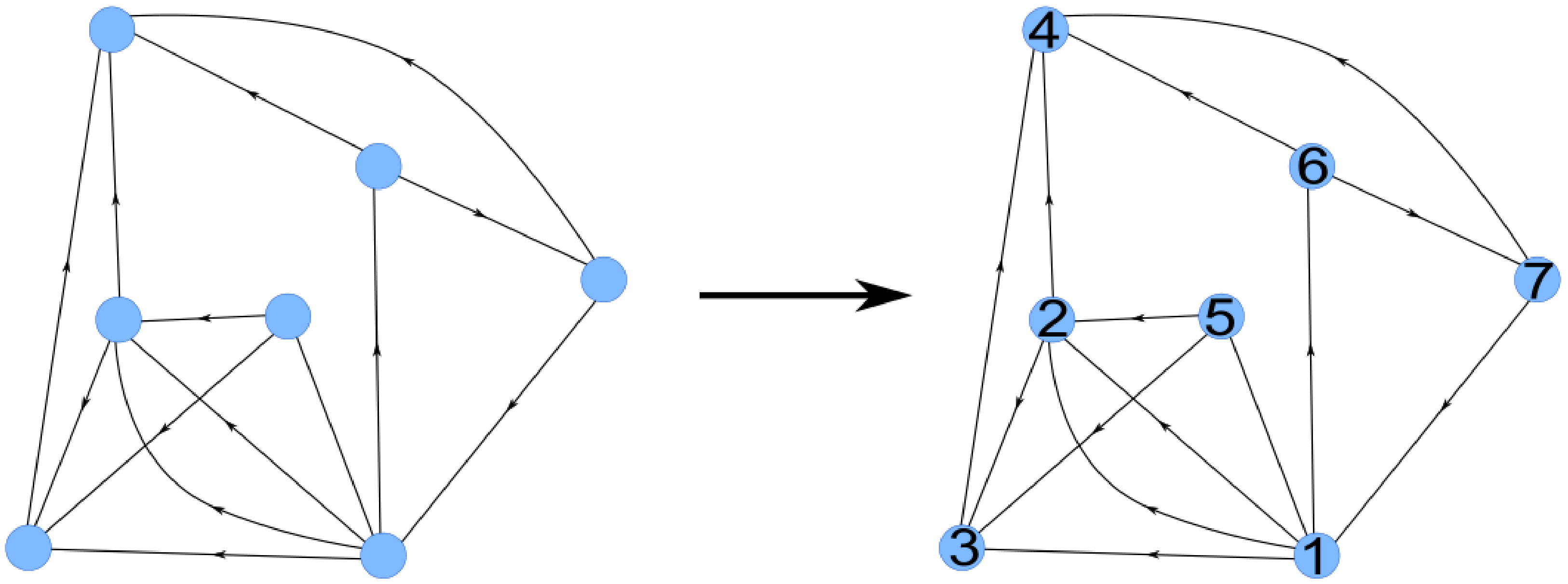
The Clique Vector: Pc (G) : # of cliques containing the vertex, in the same order of vertices of Pv (G).

d = max(valency)

Concatenate 0s if |Pv (G)| < d

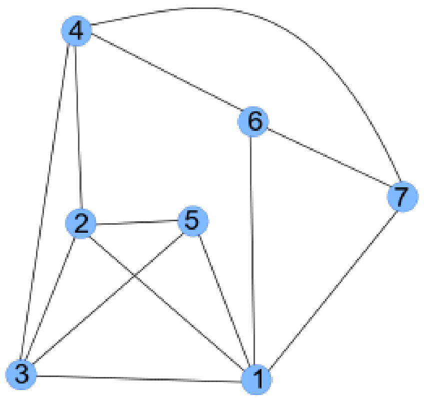
P(G) $\in R^{2d+3}$: concatenation of Pg (G), Pv (G), Pc (G).

From Graphs to a Point Cloud



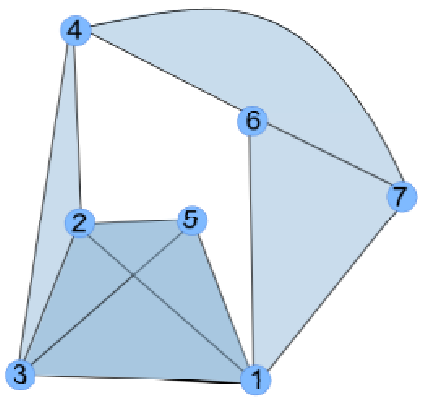
Pv (G) =⟨6,5,4,4,3,3,3⟩

From Graphs to a Point Cloud

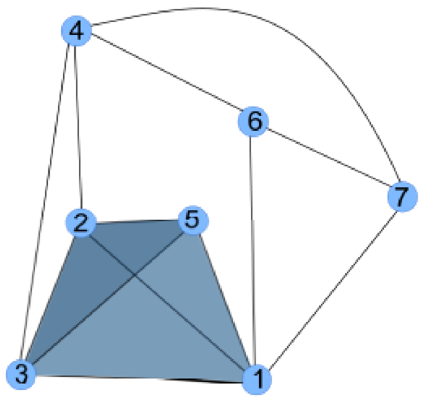


(1,1,1,1,1,1,1)

(5,4,4,4,3,3,3)



(4,4,4,2,3,2,2)



(1,1,1,0,1,0,0)

This graph has no cliques of size higher than 4.
The clique vector in this example is Pc (G)= ⟨11,10,10,7,8,6,6⟩

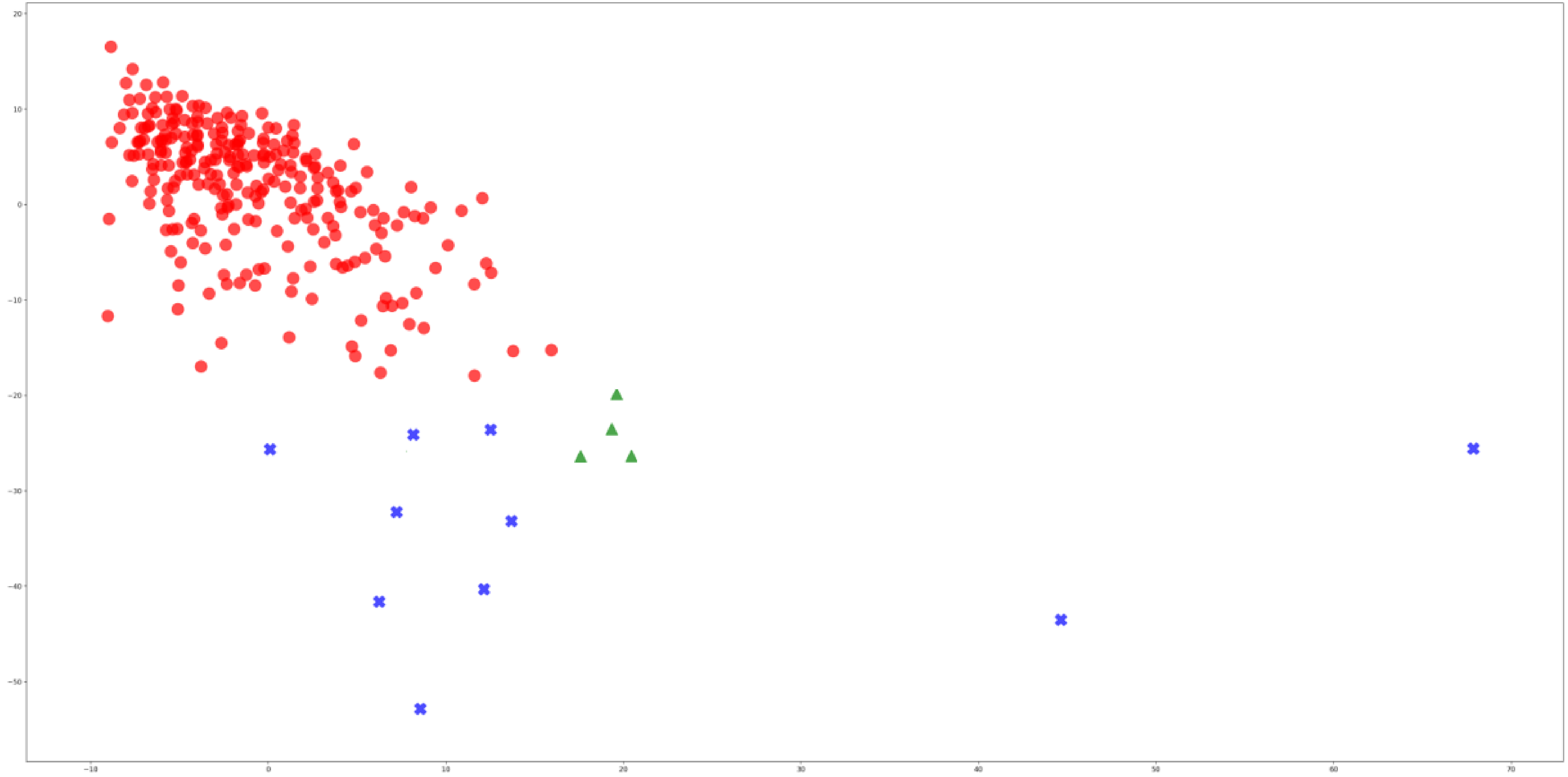
obtained by taking the sum of the 4 vectors presented under graphs.

From Graphs to a Point Cloud

The final vector < V(G), E(G), CN(G) > lives in $R^{2|V(G)|+3}$.

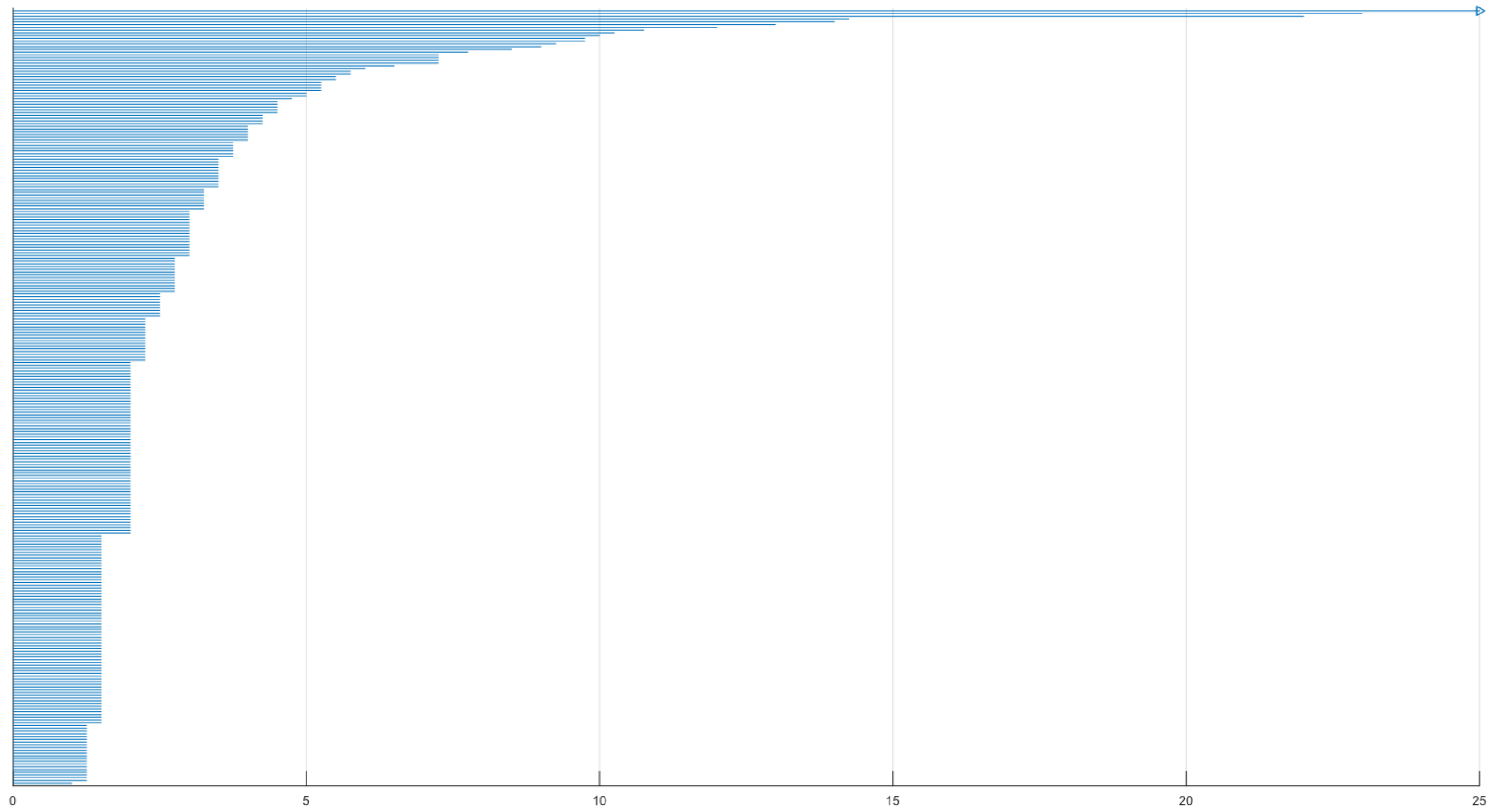
Its length depends on the graph. In order to make all of them live in the same space we augment as many zeros as necessary to the points with small dimensions.

Output



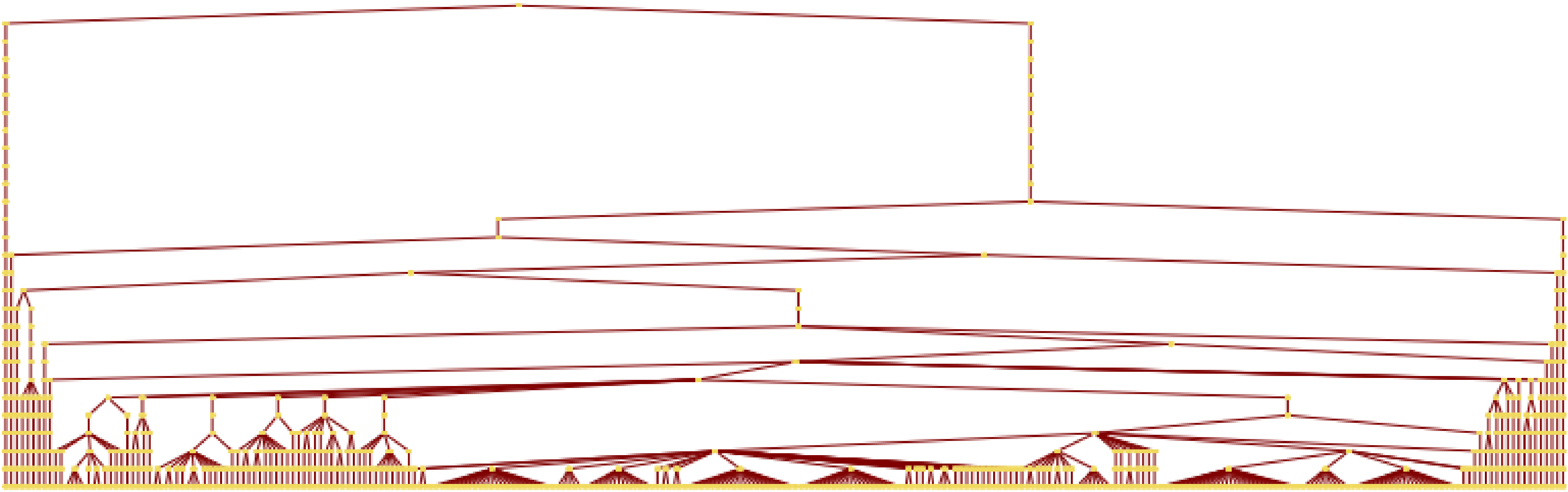
A 2d multidimensional scaling projection for the resulting point cloud.

Output



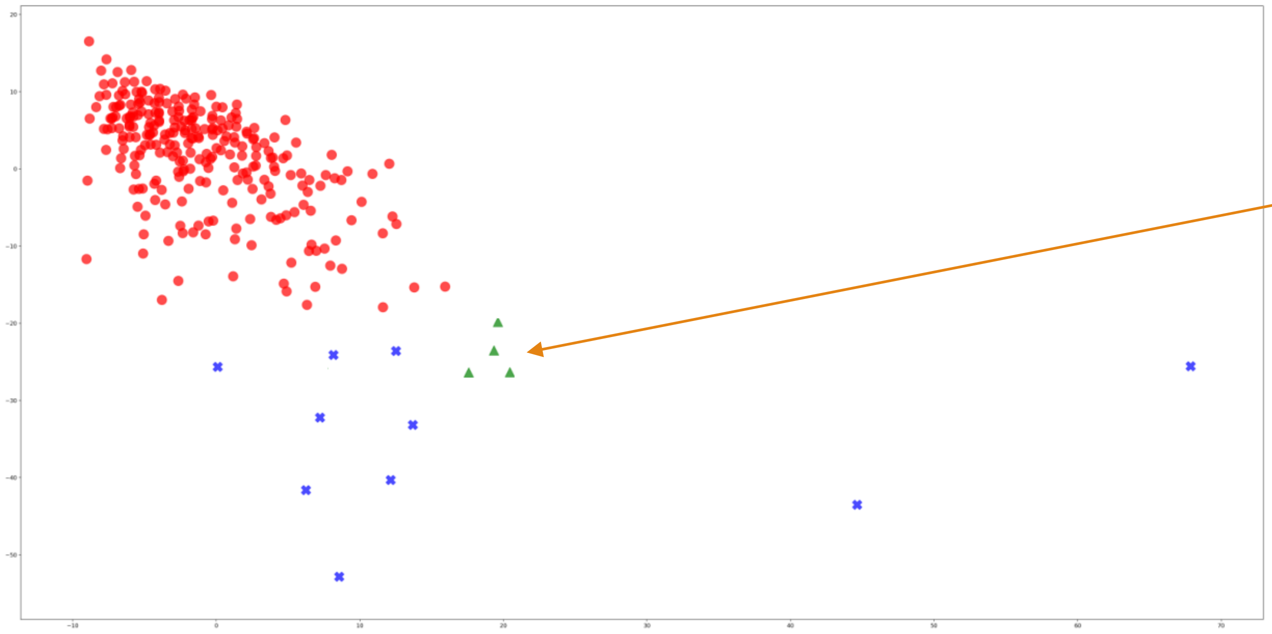
The barcode diagram describing the birth and death of the connected components.

Output



Tree dendrogram tree representing merging components at all levels in the h-clustering

Output



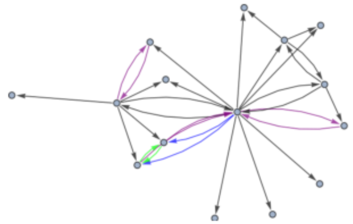


FIGURE 15. ctg7180000088928

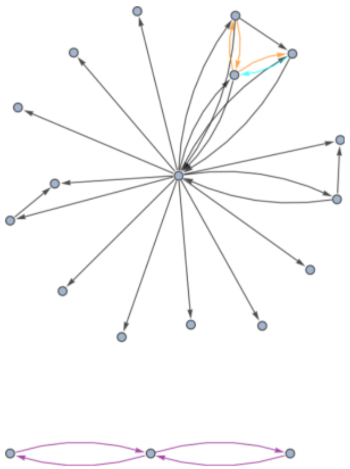


FIGURE 16. ctg7180000088096

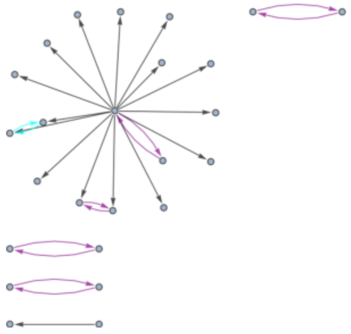


FIGURE 17. ctg7180000067742

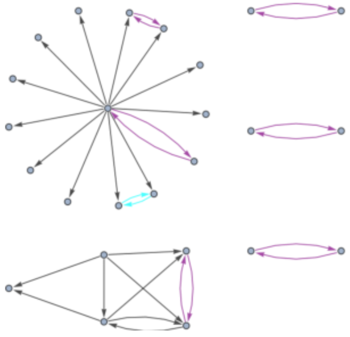


FIGURE 18. ctg7180000067187

Output

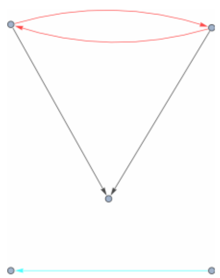


Figure : ctg7180000087289

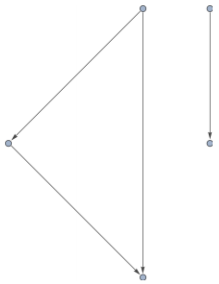
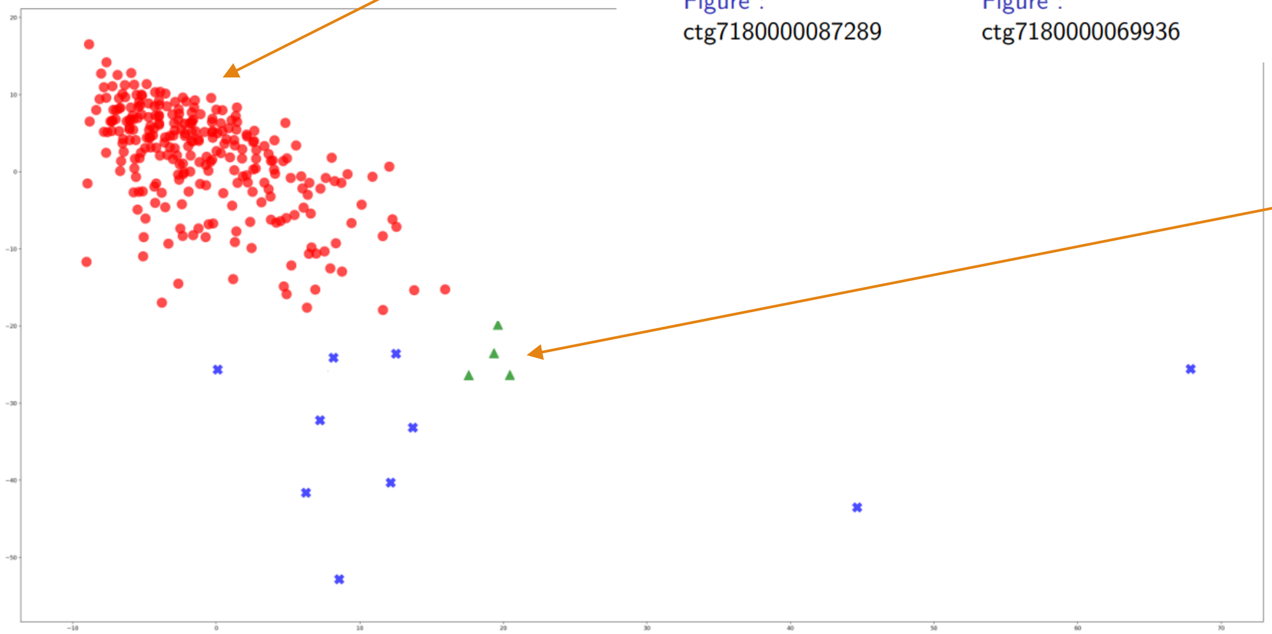


Figure : ctg7180000069936



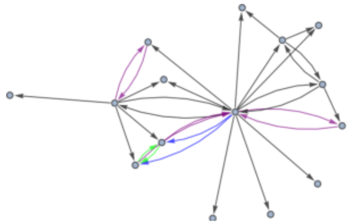


FIGURE 15. ctg7180000088928

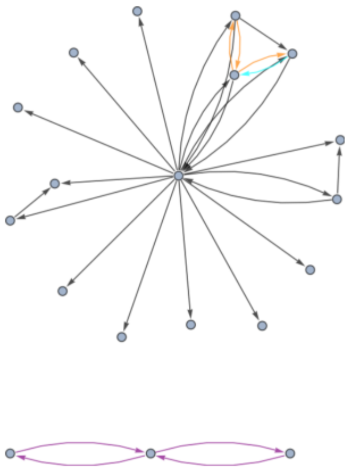


FIGURE 16. ctg7180000088096

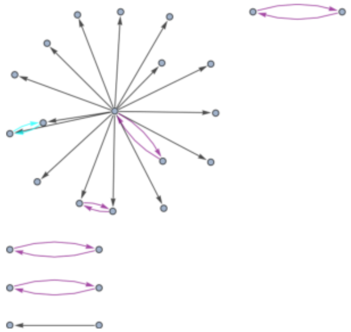


FIGURE 17. ctg7180000067742

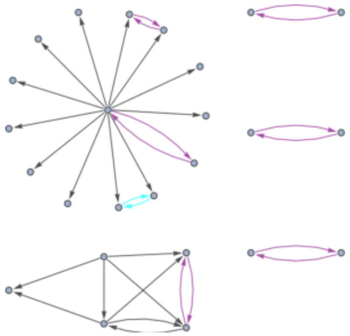
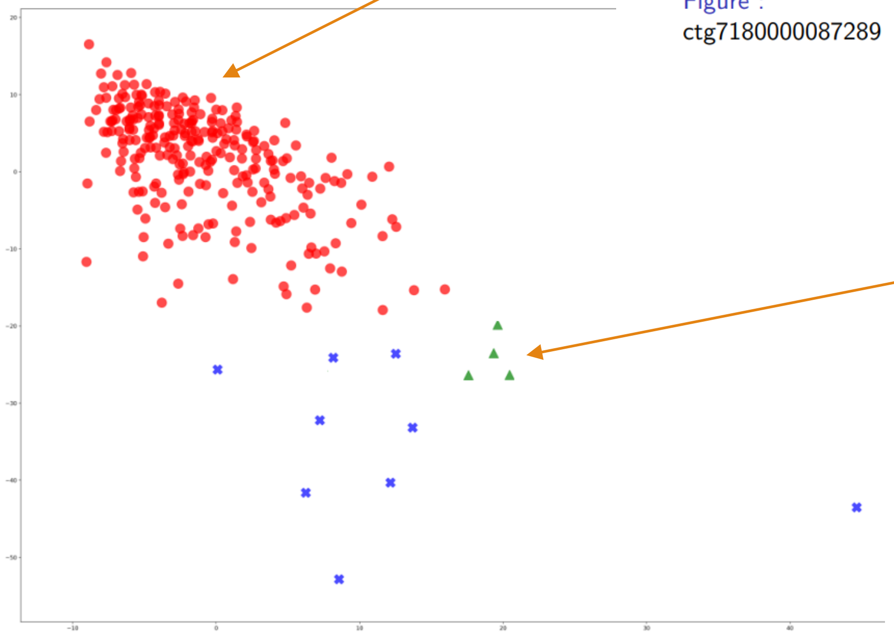


FIGURE 18. ctg7180000067187

Output



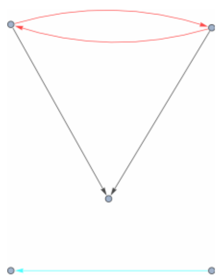


Figure : ctg7180000087289

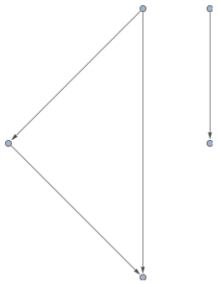


Figure : ctg7180000069936

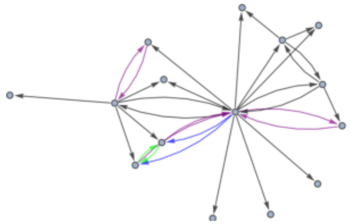


FIGURE 15. ctg7180000088928

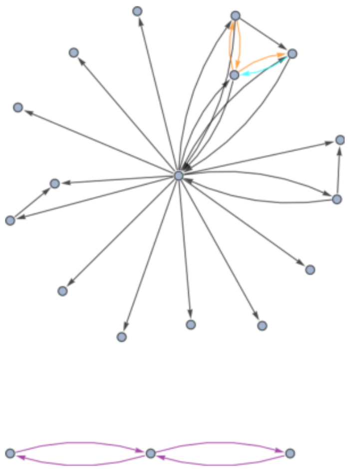


FIGURE 16. ctg7180000088096

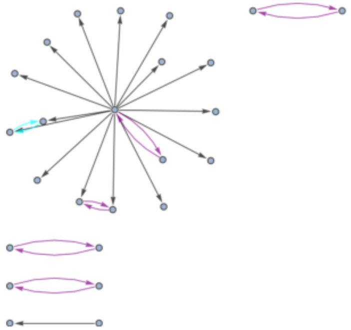


FIGURE 17. ctg7180000067742

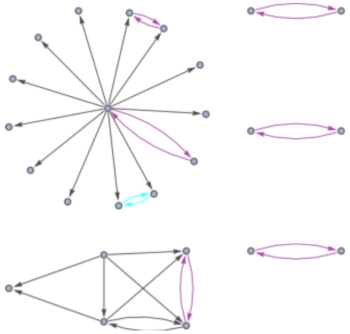


FIGURE 18. ctg7180000067187

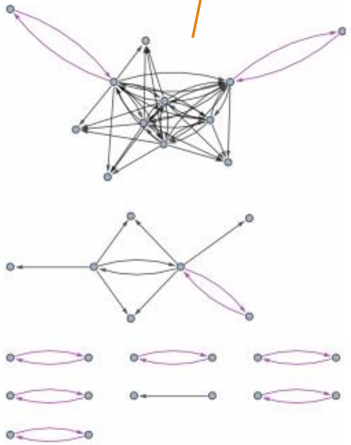
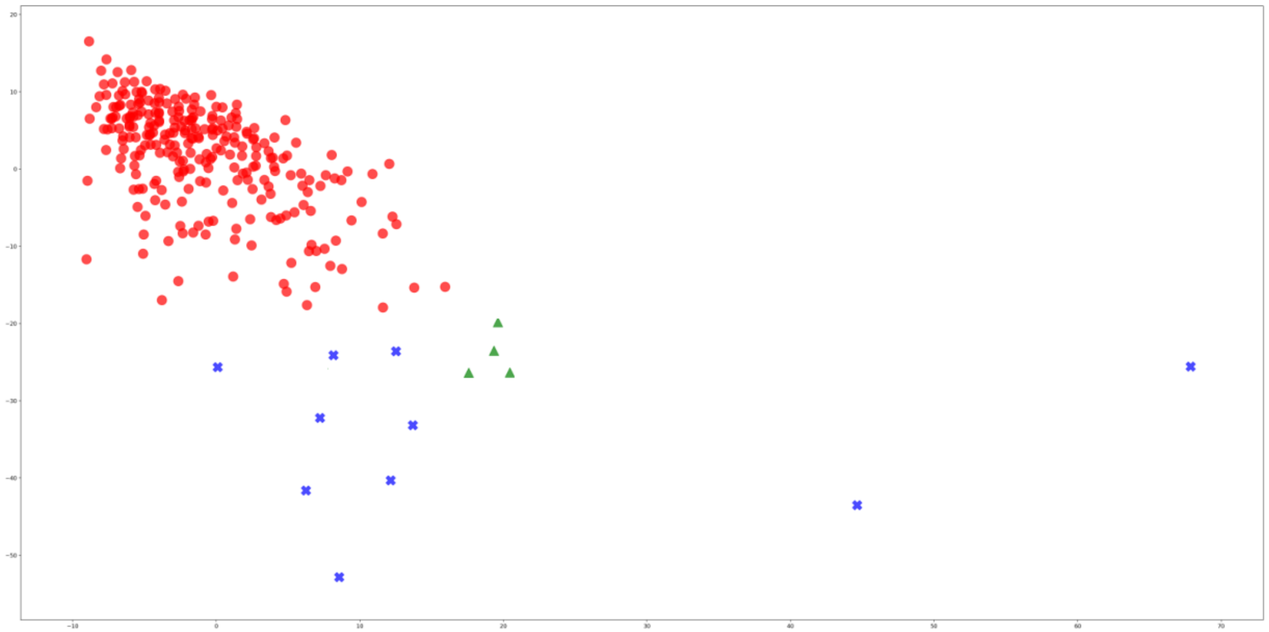


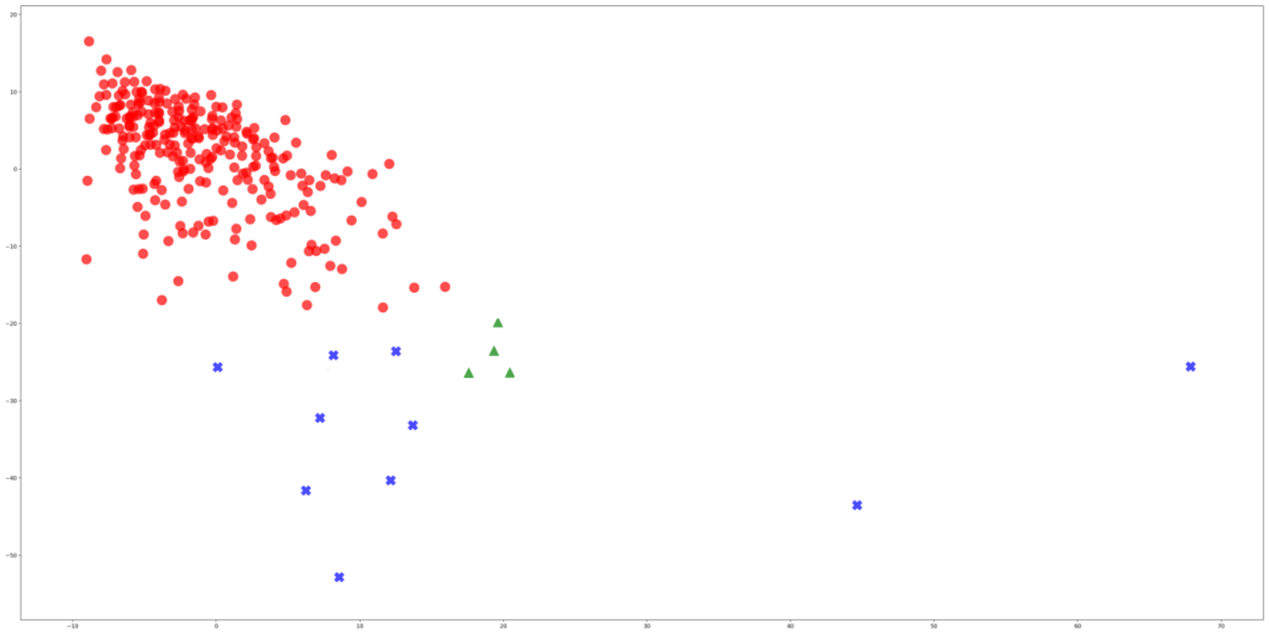
Figure : ctg7180000067223

Output



The outputs indicated that a single large cluster is formed, with some isolated singleton clusters.

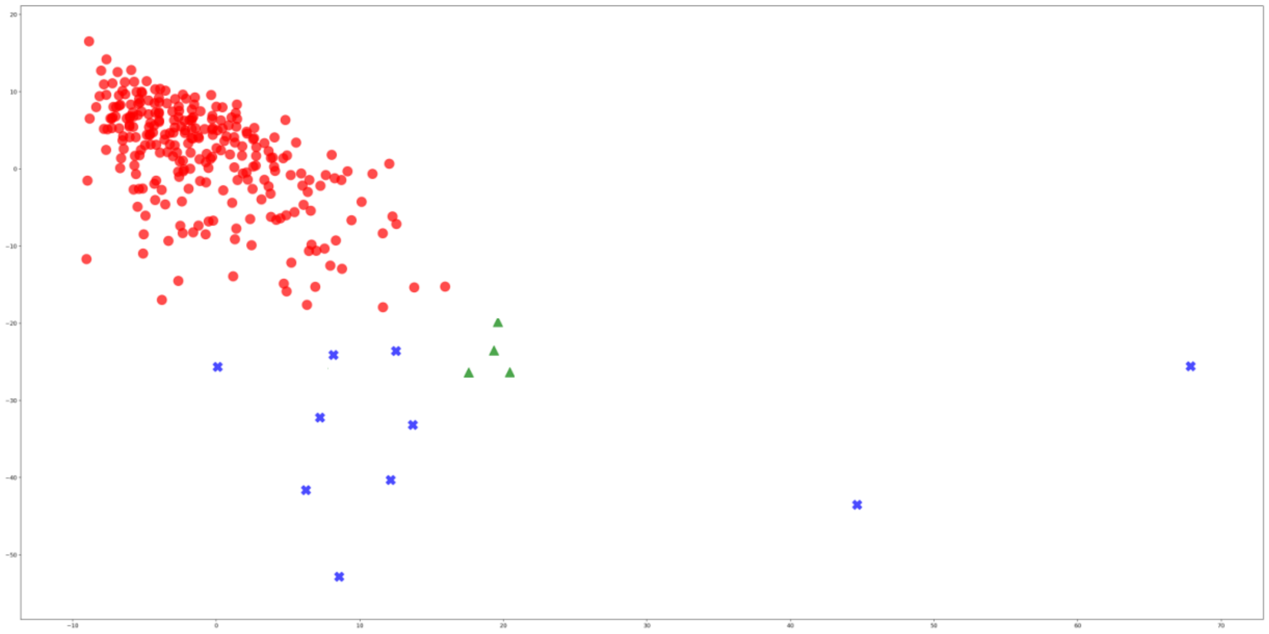
Output



The outputs indicated that a single large cluster is formed, with some isolated singleton clusters.

This is interpreted that most genes have similar and simple interaction patterns

Output



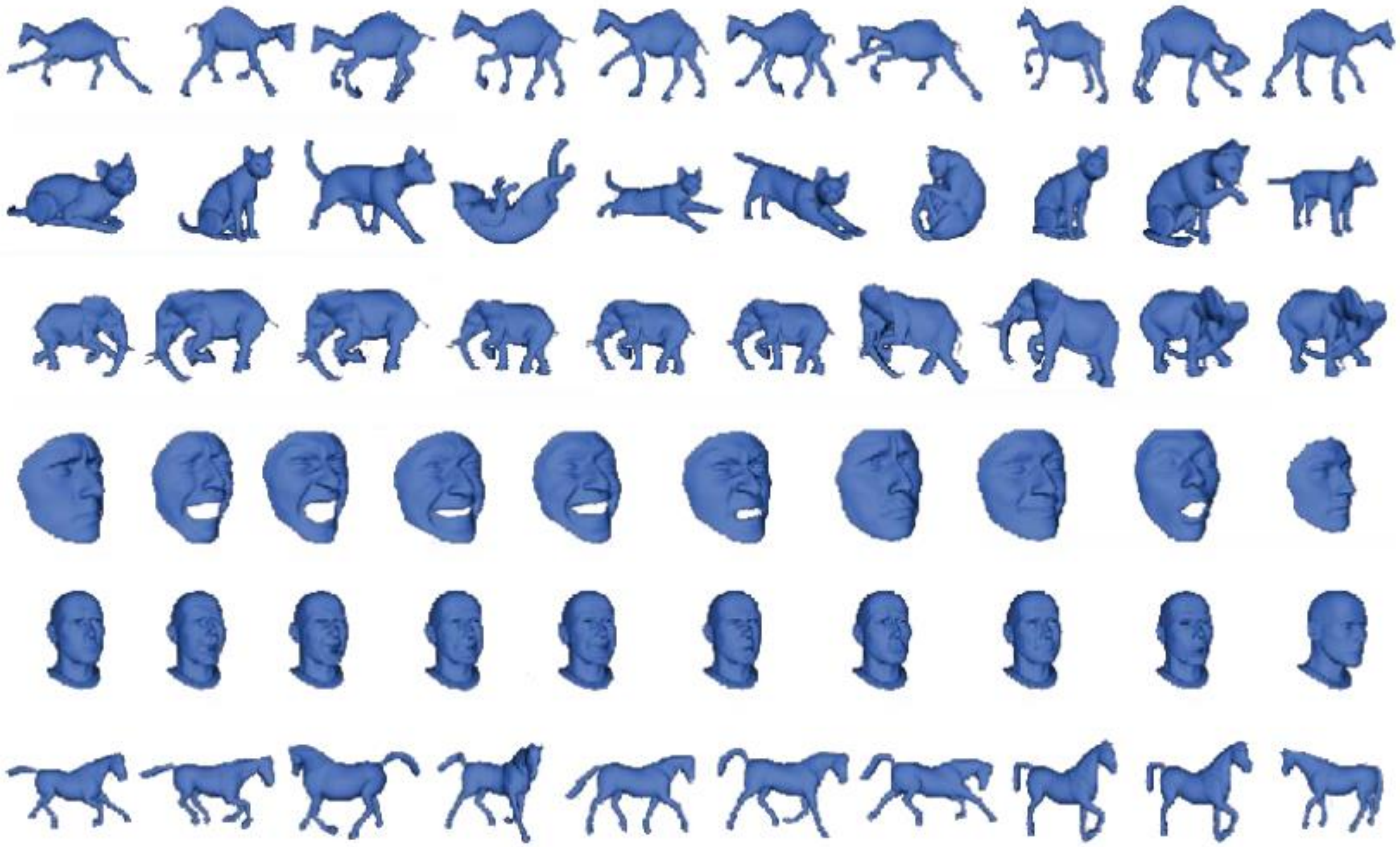
The outputs indicated that a single large cluster is formed, with some isolated singleton clusters.

This is interpreted that most genes have similar and simple interaction patterns

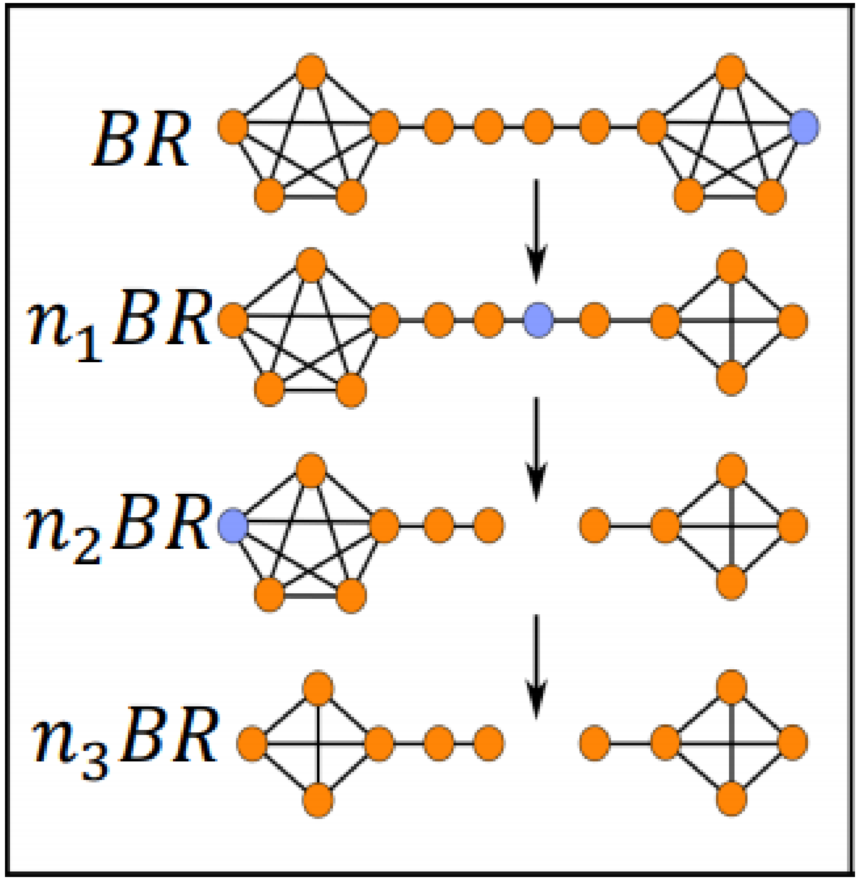
The Singleton clusters correspond to genes with complex interaction patterns that are unique and rare among the other genes.

Part II
Similarity among datasets

Detecting similarity between data sets



Detecting similarity between data sets



Measuring distance between two persistence diagrams

data

↓

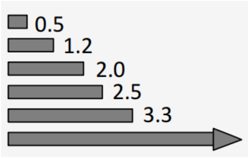
Persistence diagrams

↓

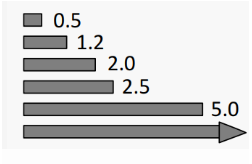
Distance between persistence diagrams

data1

data2



PD(data1)



PD(data2)

Distance between PD(data1) and PD(data2)

Measuring distance between two persistence diagrams

data

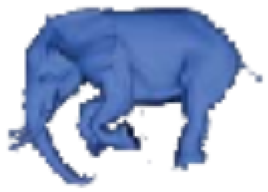
↓

Persistence diagrams

↓

Distance between persistence diagrams

data1



data2

In this example we want the distance to be larger than the previous one

PD(data1)

PD(data2)

Distance between PD(data1) and PD(data2)

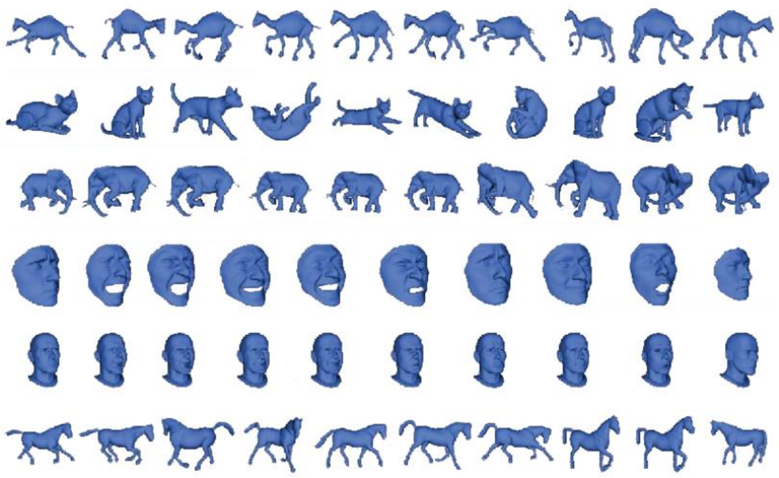
Bottleneck distance between two persistent diagrams

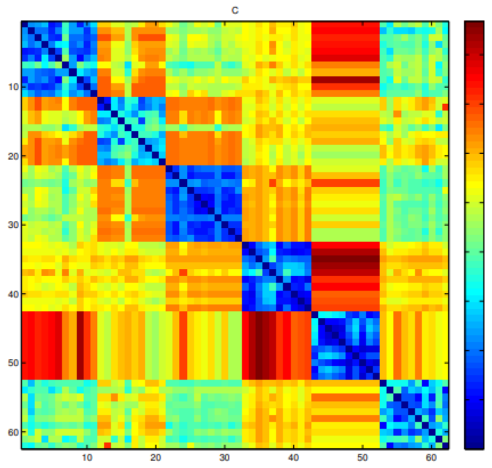
- A persistence diagram can be thought of as a summary of topological features of a given data set.
- To quantify the structural difference between two datasets D1 and D2 , we compute the bottleneck and Wasserstein distances between their persistence diagrams.
- Given two persistence diagrams X and Y, let η be a bijection between points in the diagram.

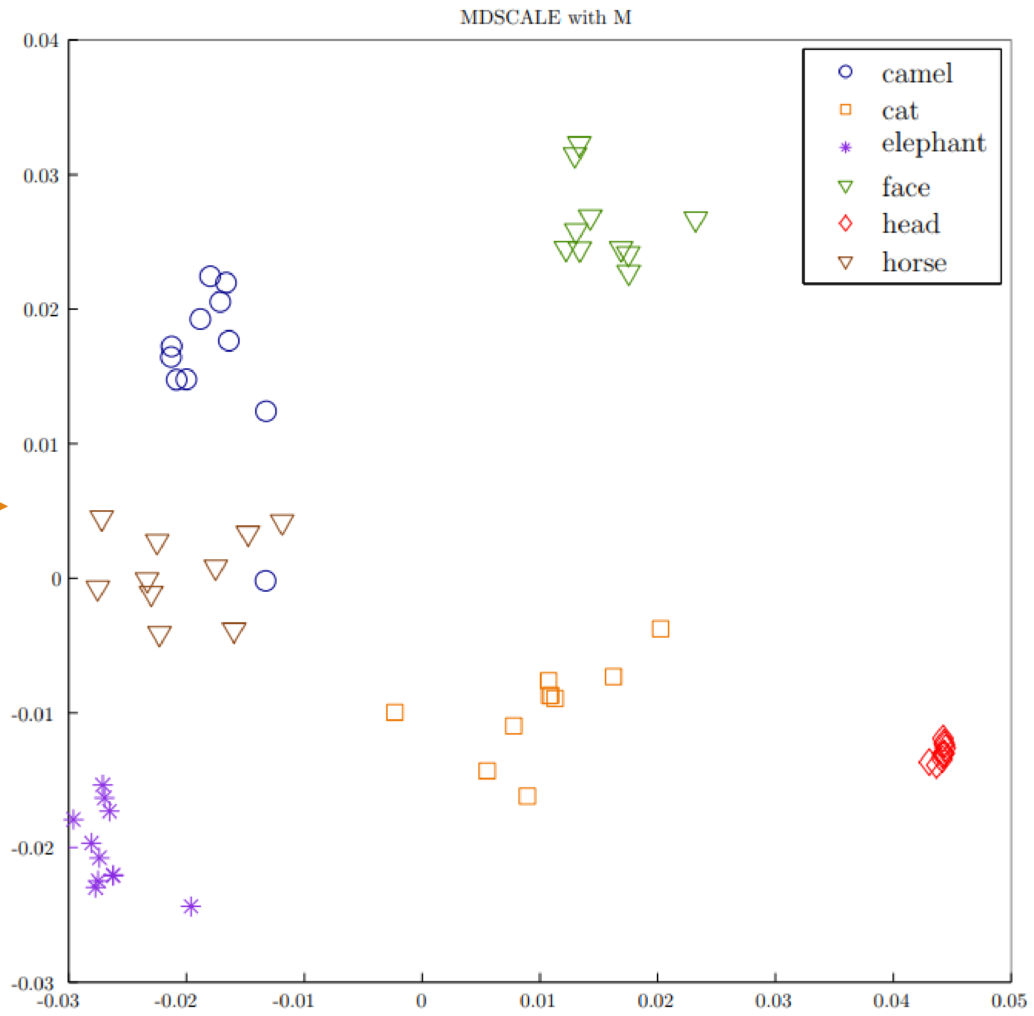
$$W_\infty(X,Y) = \inf_{\eta : X \to Y} \sup_{x \in X} \|x - \eta(x)\|_\infty$$

$$W_q(X,Y) = \left[\inf_{\eta : X \to Y} \Sigma_{x \in X} \|x - \eta(x)\|_\infty^q \right]^{1/q}$$

Bottleneck distance between two persistent diagrams







Input data

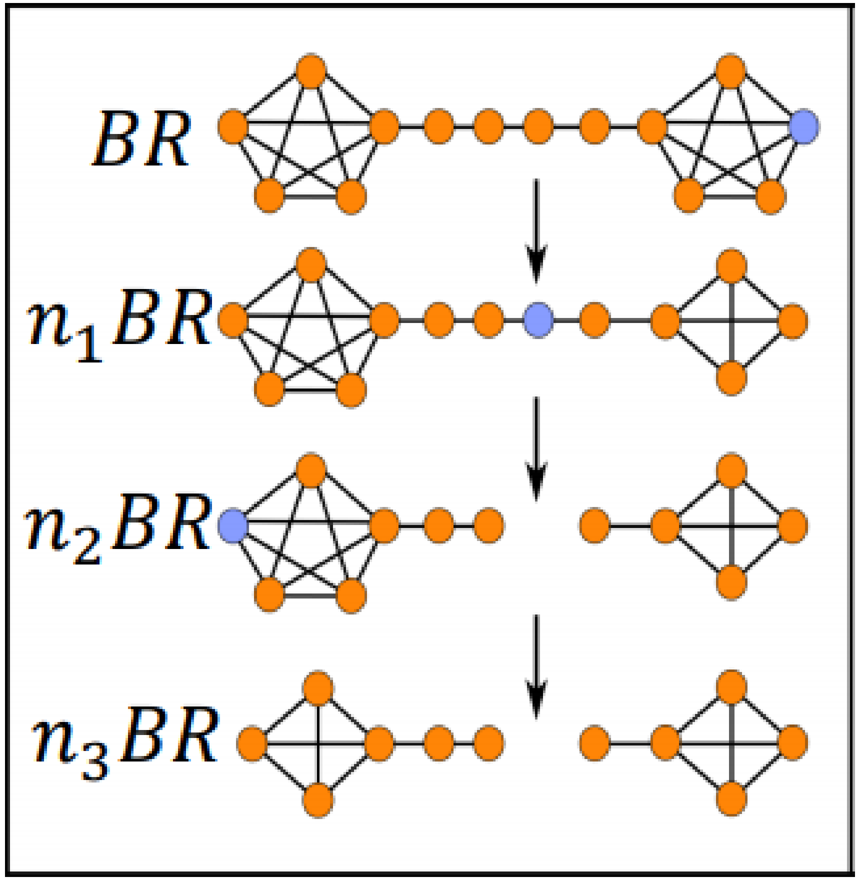
Matrix M describes the pair-wise distance between the persistence diagrams of each data element

MDS plot of the matrix M with labels corresponding to each class.

Detecting similarity between data sets-graph similarity detection

	BR	n_1BR	n_2BR	n_3BR
BR	0	0.5	0.5	0.5
n_1BR		0	0.5	0.5
n_2BR			0	0.5
n_3BR				0

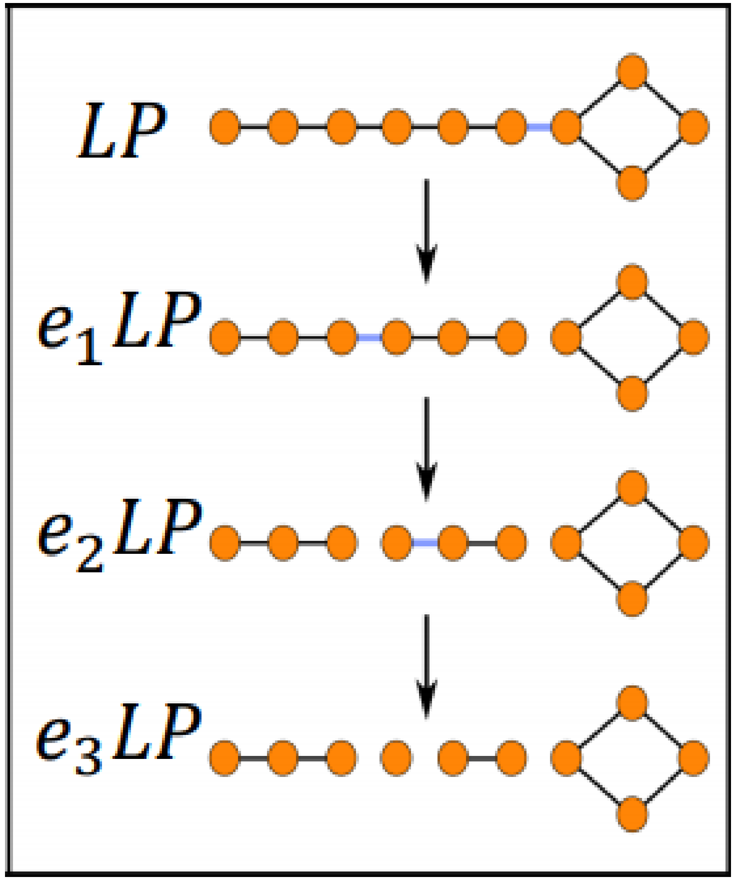
	BR	n_1BR	n_2BR	n_3BR
BR	0	0.25	0.5	1
n_1BR		0	0.5	0.25
n_2BR			0	0.25
n_3BR				0



Detecting similarity between data sets-graph similarity detection

	LP	n_1LP	n_2LP	n_3LP
LP	0	0.5	0.5	0.5
e_1LP		0	0.5	0.5
e_2LP			0	0.5
e_3LP				0

	LP	n_1LP	n_2LP	n_3LP
LP	0	0.25	0.75	1
e_1LP		0	0.5	0.75
e_2LP			0	0.25
e_3LP				0



PH Softwares

- JavaPlex : an easy to use java library
- Perseus a C++ library
- TDA : an R library

Other libraries :

- GAP Persistence
- DIPHA
- GUDHI
- Dionysus

Gudhi Library

Gudhi Library

Definition from the website : The GUDHI library is a generic open source **C++ library**, with a **Python interface**, for Topological Data Analysis (**TDA**) and Higher Dimensional Geometry Understanding.

http://gudhi.gforge.inria.fr/

Gudhi Library : rips_distance_matrix_persistence

This exe operates only on a distance matrix input file.

An example of basic usage of this exe :

rips_distance_matrix_persistence INPUTFILE -d 1 -r 5 -o out.txt

d : is the max dimension at which the rips complex computes
r : is the max distance at which we stop computing new simplicies
o : the output file

Gudhi Library : rips_persistence

This exe operates only on a distance matrix input file.

An example of basic usage of this exe :

```
rips_persistence INPUTFILE -d 1 -r 5 -o out.txt
```

d : is the max dimension at which the rips complex computes
r : is the max distance at which we stop computing new simplicies
o : the output file

Gudhi Library : bottleneck_read_file_example

This exe takes as input 2 persistence diagrams

An example of basic usage of this exe :

bottleneck_read_file_example INPUTFILE1 INPUTFILE2

each input file must contain in each line : birth death